

## Aplicação de banco de dados orientado a objetos na modelagem multidimensional

Sueli de Fatima Poppi Borba<sup>1</sup>, Aran Bey Tcholakian Morales<sup>2</sup>

<sup>1</sup>Universidade Paranaense – Unipar. Instituto Superior de Ciências Exatas, Agrárias, Tecnológicas e Geociências. Av Brasil 1123 Cianorte PR, Brasil

<sup>2</sup>Universidade Federal de Santa Catarina. Departamento de Engenharia da Produção. Trindade Florianópolis SC, Brasil

sueli@unipar.br, aran@stela.ufsc.br

**Abstract.** *The object orientation paradigm is a modeling standard to information system and the UML diagrams have been used for computation professionals. The data warehousing can use object features in order to make more flexible multidimensional modeling applications. This paper presents a methodology whose purpose is to introduce multidimensional model in object-oriented database, using UML diagrams and ODMG standard.*

**Resumo.** *O paradigma da orientação a objetos apresenta-se como um padrão para a modelagem de sistemas de informação e a representação através dos diagramas da UML são utilizados pelos profissionais da área. A utilização das propriedades de persistência de objetos pode ser utilizada em ambientes de gerenciamento, como o data warehouse, fornecendo flexibilidade na aplicação do modelo multidimensional. O presente artigo apresenta uma proposta de metodologia para implantar o modelo multidimensional em banco de dados orientado a objetos, seguindo a representação através dos diagramas da UML e o padrão da linguagem de definição de objetos da ODMG.*

### 1. Introdução

A visão multidimensional de dados não é um conceito novo, pois a necessidade de se obter de maneira rápida e simples todo o histórico de informação do ambiente operacional faz com que as organizações busquem alternativas para estruturar e acessar seus dados.

Nos últimos anos, observa-se que a comunidade técnico-científica está direcionando seus trabalhos para um paradigma que se consolida no processo conceitual de desenvolvimento de sistemas - a orientação a objetos. Portanto, aliar a tecnologia de Banco de Dados Orientados a Objetos (BDOO) à tecnologia de *Data Warehouse* - DW mostra-se como uma opção que permite o gerenciamento de dados como objetos. Porém, as atuais abordagens de modelagem de DW não apresentam mapeamentos para o tratamento dos objetos de um banco de dados, subutilizando a tecnologia da orientação a objetos.

Nesse contexto, o presente trabalho tem como objetivo propor uma metodologia para a implementação da modelagem multidimensional, representada graficamente na UML e seu mapeamento em BDOO, seguindo os padrões da orientação a objetos.

O presente artigo está organizado em 8 seções. A seção 2 relaciona as etapas propostas para a aplicação do modelo dimensional no paradigma OO. A seção 3 apresenta a etapa 1 da proposta, a seção 4 apresenta a etapa 2, seguida da seção 5 que descreve a etapa 3. Na seção 6 e 7 estão as etapas dos padrões e implementação OO, validando a proposta apresentada. Finalmente, na seção 8 são apresentadas as conclusões.

## **2. Definição da Proposta Metodológica**

A metodologia adotada nesse trabalho fundamenta-se nas seguintes tarefas:

- levantamento sobre a modelagem de dados, especificamente os trabalhos que apresentam metodologias para o modelagem dimensional;
- análise dos componentes do paradigma orientado a objetos, aplicados a modelagem dimensional;
- elaboração de um conjunto de técnicas e processos, gerando as etapas de uma metodologia para a implementação do modelo dimensional no paradigma da orientação utilizando sua representação em UML;
- validação da proposta.

A partir da análise e comparação das metodologias para a implementação do modelo dimensional (Kimball, 1997, 2002; Trujillo et al., 2003; Abelló et al., 2005), a proposta aqui apresentada busca atender os requisitos da modelagem dimensional. A metodologia proposta envolve cinco etapas, descritas nas seções subseqüentes, assim relacionadas e representadas na Figura 1: definição do modelo de negócio; geração do modelo dimensional; representação do modelo dimensional na UML; mapeamento do modelo no padrão OO e implementação do modelo em BDOO.

## **3. Etapa 1: Definir o Modelo de Negócio**

A primeira etapa é cumprida no âmbito organizacional, através da decisão das áreas a serem atendidas na modelagem multidimensional e, conseqüentemente, do DW. Nesta fase são avaliados os detalhes a respeito do modelo de negócio sobre o qual se dará o desenvolvimento do projeto, analisando-se os princípios básicos de quais requisitos de negócio serão traduzidos em objetos físicos na base de dados.

O modelo de negócio usado como exemplo para validar a proposta baseia-se no modelo de serviços financeiros, proposto como exemplo por Kimball (2002, p. 227). Esse modelo será utilizado em todas as etapas do trabalho, validando a metodologia proposta.

Nesta fase são avaliados os detalhes a respeito do modelo de negócio sobre o qual se dará o desenvolvimento do projeto, analisando-se os princípios básicos de quais requisitos de negócio serão traduzidos em objetos físicos na base de dados. São determinadas as fontes de dados utilizadas na implementação do projeto. O resultado dessa etapa consiste em uma documentação que relacione:

- finalidade do projeto;
- limite do escopo do negócio a ser modelado;
- ferramentas e aplicativos a serem utilizados.

O interesse volta-se para a observação apenas das estruturas preliminares, genéricas, verificadas a partir dos requisitos dos usuários e nos esquemas conceituais existentes.

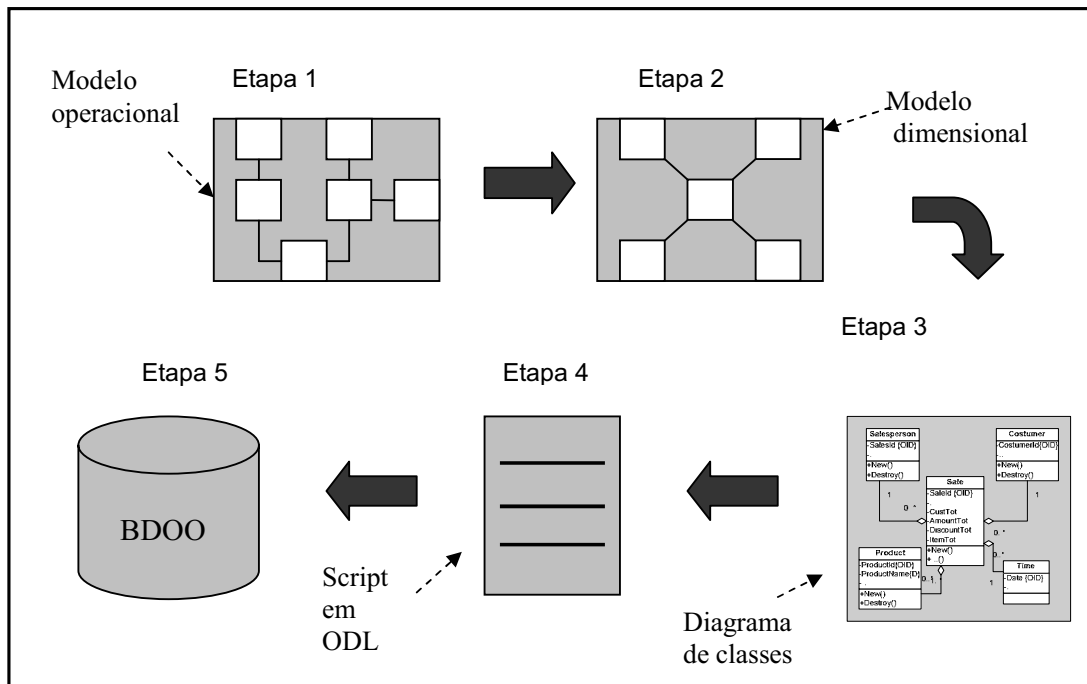


Fig. 1 - Resumo da metodologia.

#### 4. Etapa 2: Gerar o Modelo Dimensional

Na etapa 2 da metodologia são identificadas as bases transacionais com suas respectivas funcionalidades no meio operacional, que originam o modelo dimensional.

O modelo dimensional é padronizado em sua representação pelos esquemas *Star* (Estrela), *Snowflake* (Floco de neve) ou variações desses modelos. A composição típica do modelo Estrela possui uma tabela central denominada Fatos (fact table) e um conjunto de entidades denominadas Dimensão (dimension tables), na forma de uma estrela.

Partindo do modelo de negócios existente na organização, projetado através do modelo ER (Entidade-Relacionamento), analisam-se as dimensões que podem vir a fazer parte do processo, e como podem ser agregados os valores detectados. Inicialmente, as dimensões são consideradas de maneira geral, para posterior detalhamento de atributos e suas hierarquias.

Para a geração do modelo dimensional, a proposta baseia-se no padrão de ambiente de DW, que é gerado a partir do ambiente operacional, com o modelo de negócio de Serviços Financeiros.

Deve existir a correlação entre os esquemas conceituais do banco de dados operacional, observando os atributos das entidades e relacionamentos que identificam fatos em um esquema ER. Essa análise pode indicar dimensões implícitas no negócio a ser modelado. Para gerar o modelo dimensional, portanto, faz-se necessário desmontar hierarquias, que são os relacionamentos normalizados do modelo ER, anexando as informações relevantes às tabelas Fatos ou Dimensão correspondentes.

Essa fase da metodologia caracteriza-se pela definição dos aspectos iniciais do modelo de negócio, que correspondem à definição da granularidade, dimensões e fatos. Também são estabelecidas as medidas de derivação.

#### **4.1. Granularidade**

A granularidade do fato é o nível de detalhe com que são gravados os dados. Para que possam ser efetivamente analisados, os dados devem estar no mesmo nível de granularidade. Como uma regra geral, os dados podem ser mantidos com o maior nível de detalhamento e, posteriormente, são sumarizados, gerando um nível mais baixo de granularidade.

Modelos dimensionais que expressam dados no mais baixo nível de detalhe geram flexibilidade e extensibilidade máxima. Os dados atômicos podem ser resumidos de qualquer modo. Igualmente, os dados atômicos podem ser estendidos com atributos adicionais, medidas ou dimensões sem romper com processos existentes (Kimball, 2002).

Observando a questão da granularidade, pode-se considerar, também a aditividade, em que medidas podem ser resumidas. A aditividade torna-se importante quando ocorre a possibilidade de sumarização em tabelas de fatos. Geralmente é desejável que as medidas possam ser completamente aditivas, pois quando não são, deve-se considerar rompê-las em seus elementos atômicos.

Observando o modelo de negócio, os esquemas operacionais indicam os Fatos do modelo dimensional das transações e posição da conta bancária. São analisadas as hierarquias e, conseqüentemente, a necessidade de se anexar seus dados às tabelas Fatos ou Dimensão correspondentes.

Como regra geral, os dados podem ser mantidos com o maior nível de detalhamento e posteriormente, sumarizados, gerando um nível mais baixo de granularidade, oferecendo flexibilidade às consultas do usuário. O modelo proposto permite a geração do mais baixo nível de granularidade, considerando que apresenta o maior nível de detalhamento, derivado do ambiente operacional. Porém, será utilizada a aditividade para sumarizar os dados na tabela Fatos, visando atender os objetivos do modelo.

#### **4.2. Dimensões e Fatos**

A modelagem multidimensional apresenta os seguintes elementos básicos (Kimball, 2002):

- Fatos: coleção de itens de dados, que se compõem de dados de medida e de contexto, normalmente representados por dados numéricos e que são o foco da investigação do suporte à decisão, sendo considerada a principal tabela de um modelo dimensional;

- Dimensões: elementos que participam de um fato, normalmente não possuem atributos numéricos e constituem-se de agrupamentos lógicos de atributos com uma chave de relacionamento comum;
- Medidas: atributos numéricos que representam um fato, normalmente qualificadores métricos conceituais.

O contexto de *data warehousing* baseia-se na equivalência de tempo, considerando a perspectiva de análise de negócio, agregando dados por dia, semana, mês ou ano, por exemplo. Portanto, se o próprio negócio que está sendo analisado no DW já possui esse parâmetro, deve-se incluí-lo como atributo de tempo, caso contrário, faz-se necessária a inserção de um atributo com a característica de preservar o tempo da ação no contexto organizacional.

No modelo proposto toda medida descreve exatamente o mesmo grupo de dimensões, gerando, portanto, o fato Posição Conta. Portanto, a entidade de transação é definida, ou seja, a tabela Fatos, com base na posição financeira das contas de clientes.

As entidades componentes são representadas pelas entidades que possuem um relacionamento um-para-muitos com a entidade de transação, que no modelo são geradas a partir das entidades Agencia e Gerente. A entidade de classificação do modelo também caracteriza relacionamento 1:n. Segundo o modelo proposto, esses relacionamentos são identificados entre as entidades Conta, Saldo e Transação. A tabela Fatos é gerada a partir do relacionamento dessas entidades operacionais, que possuem os principais atributos para a identificação do negócio a ser modelado. Considerada a questão central a ser analisada, a entidade de classificação é gerada a partir das entidades Transação e Saldo, enquanto a entidade Conta gera uma entidade componente.

Em uma complementação das entidades componentes e de classificação, são definidos os atributos de cada uma dessas dimensões. Os atributos operacionais que não são importantes para a análise do negócio podem ser descartados.

Lembrando a importância do fator tempo para o modelo dimensional, esse é considerado como o período em que ocorreram as transações, no exemplo, determinada data. A dimensão tempo será armazenada em dias considerando o mais baixo nível de granularidade para o atributo de data.

### **4.3. Medidas de Derivação**

Nesse momento, as informações já definidas são consideradas e inicia-se o processo de refinamento e detalhamento do modelo. Podem ser detectados atributos novos e atributos desnecessários podem ser eliminados.

Consultas em tempo de execução geralmente não são satisfatórias e é necessária a materialização de dados agregados. Portanto, propõe-se criar mecanismos para sumarizar informações, evitando a necessidade de execução do processo de sumarização em toda consulta, através de regras estabelecidas na metodologia proposta:

- para a sumarização dos dados devem ser utilizadas as funções (em SQL) *SUM*, *AVG*, *COUNT* ou equivalentes;
- quando não existir a necessidade de análise atômica no dado, esse deve ser sumarizado, caso contrário, deve-se manter o mais baixo nível de granularidade

ou manter na tabela Fatos os dados sumarizados e uma hierarquia (ou similar) com os dados atômicos.

O quadro 1 mostra algumas das medidas derivadas: SaldoMedioMensal, NumeroTransacoes, JurosPagos e TotalCredito, e suas regras de derivação, geradas a partir das tabelas de dimensão e implementadas na tabela Fatos. As medidas derivadas referenciadas no exemplo são sumarizadas através da utilização dos operadores *SUM*, *AVG* e *COUNT* na agregação dos valores de medidas de dimensões.

Quadro 1. Regra de derivação e medida derivada.

Atributo(s) origem	Regra de derivação	Medida derivada
SaldoConta.ValorSaldo	AVG(ValorSaldo) no mês	SaldoMedioMensal
Transação.NumeroTransacao	COUNT(NumeroTransacao) na data	NumeroTransacoes
Transação.ValorLancamento	SUM(ValorLancamento) para tipo = juros pagos	JurosPagos
Transação.ValorLancamento	SUM(ValorLancamento) para tipodebitocredito = credito	TotalCredito

#### 4.4. Relacionamentos

O relacionamento entre as entidades no modelo dimensional normalmente é de cardinalidade um-para-muitos (1:n), considerando que os identificadores das dimensões são atributos nos Fatos. Porém, relacionamentos muitos-para-muitos (n:n) também são possíveis para a representação do negócio modelado. Quando a cardinalidade de uma dimensão com fatos é n:n devem ser analisadas e consideradas as possíveis soluções para o problema. Kimball (1997) propõe a criação de uma tabela *bridge* entre Dimensão e Fatos.

O próximo passo é definir e refinar as hierarquias obtidas das dimensões previamente detectadas. Esse refinamento consiste em definir propriedades para os atributos das hierarquias, analisando a possibilidade da conversão em atributos descritores.

Aplicando a metodologia proposta, a etapa 2 é concluída com as definições abaixo relacionadas:

- estabelecida a granularidade do modelo, identificando a necessidade do nível sumarizado;
- gerada a tabela Fatos, a partir do escopo do problema de Transações Financeiras;
- identificadas as dimensões não associativas, que estão ligadas aos fatos, com respectivos atributos, que são representadas pelas dimensões Agencia e Gerente;
- identificada a dimensão associativa, e respectivos atributos dimensionais, representada pela dimensão Conta;
- gerada a dimensão Tempo, a partir do período da transação da operação, com sua respectiva granularidade, observando a necessidade de análise do negócio;

- determinados os atributos das entidades que serão utilizados nas várias dimensões geradas, considerando o negócio em análise;
- definidas as medidas e regras de derivação, relacionadas aos valores e números de transações;
- verificadas as hierarquias de classificação, definindo se essas serão agregadas às dimensões associativas ou estabelecer os DAGs, gerando o modelo *Snowflake*;
- verificada a existência de valores nulos, que não atendam aos requisitos de exatidão e completeza estabelecidos para o modelo;
- gerada a tabela *bridge*, no relacionamento n:n de associação entre Conta-Cliente e entre Conta-Produto;
- comparados e revisados os relacionamentos gerados no modelo dimensional com o modelo operacional original;
- verificada a necessidade da utilização dos atributos para se preservar a análise do negócio.

Portanto, o modelo dimensional proposto, gerado a partir do modelo operacional ER de Pedidos de clientes é representado na figura 2.

### **5. Etapa 3: Gerar o Modelo Dimensional Representado pela UML**

Nessa etapa o modelo dimensional está definido, representado pelo esquema *Star*, *Snowflake* ou similar. Portanto, o modelo dimensional é mapeado para o diagrama de classes, representando o modelo estático dimensional segundo o paradigma orientado a objetos, em que as dimensões e fatos são representados por classes Dimensão e Fatos.

Um diagrama de classes é um diagrama estrutural ou estático com o qual se modela a estrutura de um sistema de classes e que sob vários aspectos, assemelham-se a diagramas ER (Muller, 2002). Os diagramas de classe da UML 2.0 mostram as classes do sistema, os relacionamentos (inclusive herança, agregação e associação), operações e atributos das classes.

#### **5.1. Classes**

A partir do modelo dimensional projetado, as dimensões e fatos são representados pelas classes Dimensão e Fatos, respectivamente, sendo que a classe Fatos representa os fatos e suas medidas, definidas como atributos nestas classes.

#### **5.2. Relacionamentos**

A flexibilidade da agregação compartilhada na UML permite representar o relacionamento n:n entre a classe Fatos e classes Dimensão, registrando que uma instância de um objeto da classe Fatos pode ser relacionada a uma ou mais instâncias do objeto das Dimensões. Os BDOOs não suportam associações n-árias como relacionamentos de primeira classe, e com isso o modelo de objetos do padrão ODMG - *Object Data Management Group* (ODMG, 2005) não oferece uma maneira de declarar tais associações, sendo que a solução desse problema consiste na criação de classes de associação.

Classes Dimensão que possuem características semelhantes podem ser modeladas através do relacionamento do tipo generalização/especialização, usando os conceitos de herança da orientação a objetos.

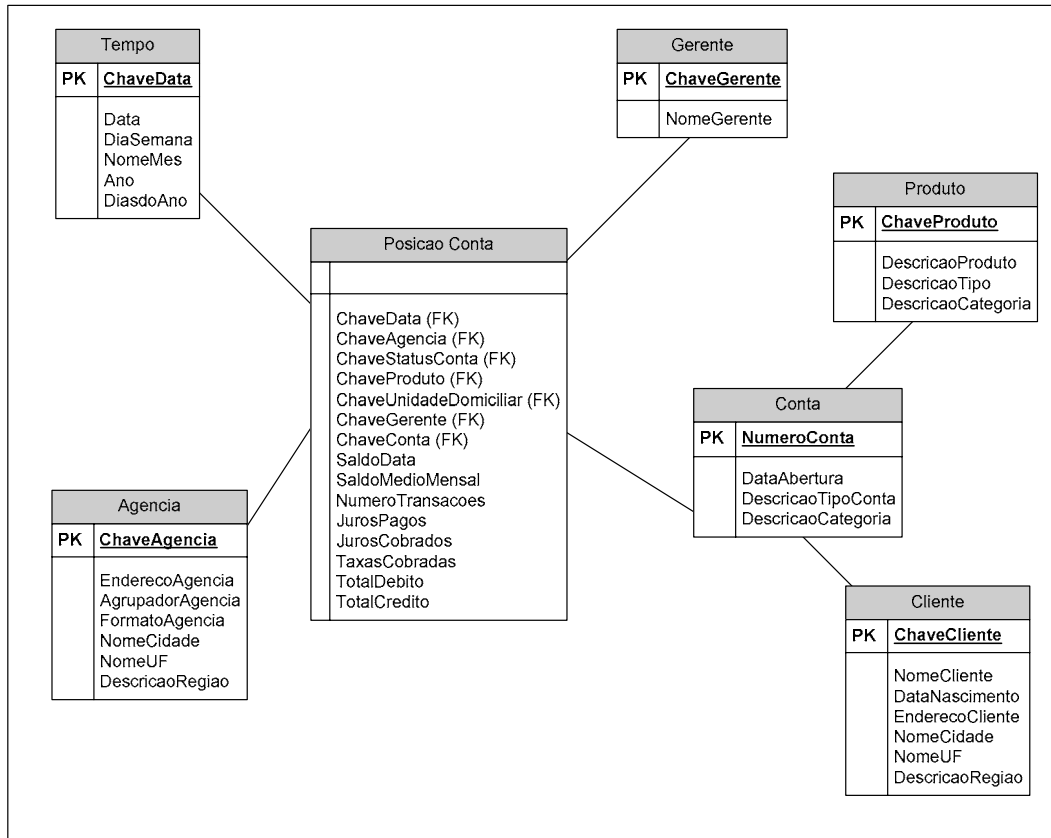


Fig 2. Modelo dimensional de Posição de Conta gerado a partir do modelo ER.

### 5.3. Identificadores e Descritores

Quando se cria um objeto, ele recebe um identificador (OID), que deve ser único para cada objeto (Nassu e Setzer, 1999). Esse identificador é gerado quando o objeto é criado, tornando-se permanente.

Além dos identificadores de objeto, podem ser associados descritores aos objetos. Um descritor é um atributo de um nível, usado para selecionar suas instâncias (Abelló, 2005).

Esses atributos descritores, com a restrição {D} especificada, além de serem utilizados na fase de análise de dados pelas ferramentas OLAP, definidos em cada um dos níveis da hierarquia de classificação no diagrama, também podem descrever cada instância do objeto.

Os descritores são identificados com a restrição {D} especificada na representação do modelo, conforme mostra a figura 3.

### 5.4. Refinar o modelo

O modelo operacional pode dar origem a dimensões grandes ou complexas e o processo simples de desnormalização pode gerar grande redundância e, conseqüentemente,

problemas de manutenção. A normalização parcial da dimensão pode ser uma contraposição entre performance e redundância, sendo necessária a validação dos dados.

Portanto, para finalizar o modelo dimensional, deve-se rever o negócio a ser modelado, verificando se os requisitos foram devidamente cumpridos.

#### **6. Etapa 4: Mapear o Modelo UML para BDOO**

O projeto de banco de dados, geralmente inclui as fases: conceitual, lógica e física. A desnormalização pode ser separada dessas fases porque envolve aspectos que não estão puramente relacionados nem ao projeto lógico nem ao físico. O processo de desnormalização deve ser implementado entre o mapeamento do modelo de dados e o projeto físico e com isto, integrado com essas etapas (Shin e Sanders, 2005).

Os relacionamentos entre os níveis conceitual e lógico, e entre o lógico e o físico são representados explicitamente pela especificação do mapeamento entre os objetos correspondentes dos diferentes níveis.

O mapeamento de um modelo de dados UML para um esquema orientado a objetos poderia ser direto se os fornecedores de sistemas de banco de dados orientado a objetos aderissem a um padrão de definição de esquemas. O padrão ODMG abrange a linguagem de consulta OQL e os recursos básicos que um BDOO deve possuir, não oferecendo interoperabilidade. Porém, considerando-o como o atual padrão existente (Turowski, 2000; Muller, 2002), as especificações dessa etapa serão definidas observando os padrões estabelecidos pela ODMG.

A ODMG definiu uma linguagem de especificação usada para representar objetos em sistemas de banco de dados. Essa linguagem de especificação independe de linguagem de programação e é usada para definir o esquema, operações e o estado dos objetos em bancos de dados. A ODL - *Object Definition Language* - é uma linguagem para definir as especificações dos tipos de objetos de acordo com o modelo de objetos ODMG (ODMG, 2005; Cattell et al., 2000).

Os SGBDs usam a linguagem de definição de dados – DDL<sup>1</sup> – e a linguagem de manipulação de dados – DML<sup>2</sup>. A DDL permite aos usuários definir os tipos de dados enquanto a DML permite criar, apagar, ler e alterar instâncias dos tipos de dados. A ODL é a DDL para tipos objetos, definindo as características dos tipos, incluindo suas propriedades e operações (ODMG, 2005).

##### **6.1. Padrão ODMG**

O modelo de objetos do ODMG 3.0 tem dois tipos de propriedades para representar o estado do objeto: o atributo, que define o estado de um tipo, e o relacionamento entre objetos, que devem possuir instâncias que podem ser referenciadas pelo identificador de objetos (Cattell et al., 2000).

A especificação do relacionamento nomeia e define um caminho através de um relacionamento. Porém, a definição do relacionamento referencia-se como a parte estática de cada objeto, não suportando a semântica comportamental que é essencial em relacionamentos todo-parte (Lee et al., 2005).

---

<sup>1</sup> DDL – *Data Definition Language*

<sup>2</sup> DML – *Data Manipulation Language*

Em ODL, classes são declaradas através de uma interface (usando a palavra chave interface); cada classe é uma coleção de atributos, que são primitivos (de um tipo de dados básico) ou relacionamento (seu valor é um objeto ou conjunto de objetos de uma determinada classe). A interface inclui: um nome, uma chave (opcional), uma *extent* e as propriedades e operações. Uma chave é um conjunto de propriedades (atributos ou relacionamentos) cujos valores são únicos para cada objeto na extensão (chaves compostas são chamadas de combinação na ODMG). A extensão é o nome do conjunto de objetos abaixo da declaração (Badia, 2005).

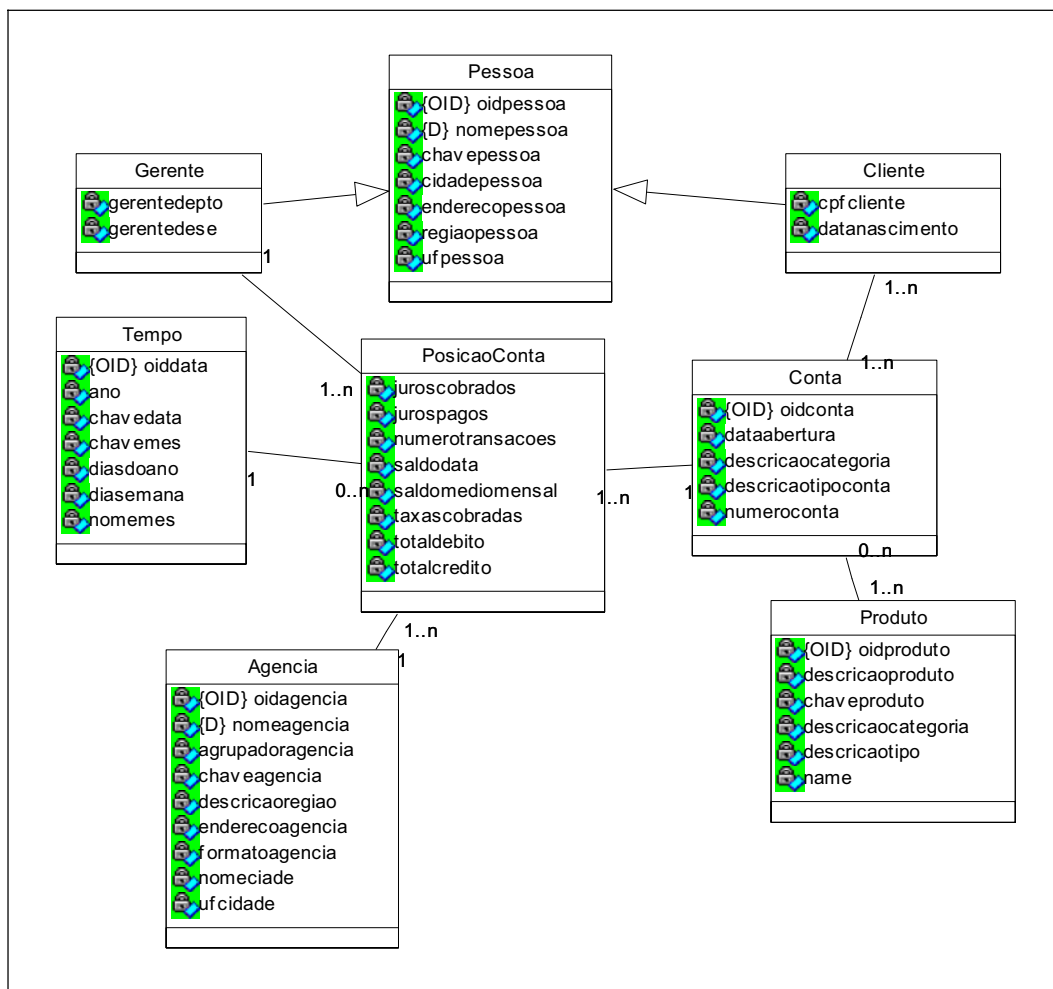


Fig. 3. Modelo dimensional PosicaoConta

Valores de atributos são tipicamente literais (atômico ou complexo), mas podem ser OIDs. Valores de relacionamentos são sempre nomes de objeto ou uma coleção aplicada para nomes de objetos. No modelo ODMG apenas relacionamentos binários são representados explicitamente, através de um par de referências inversas (usando a palavra chave inverse) (Badia, 2005; ODMG, 2005).

Valores de literais podem ser simples ou complexos. Há três tipos de literais: atômico, coleção e estruturado. Literais atômicos correspondem a tipos de dados básicos: *long*, *short*, *unsigned long*, *unsigned short*, *float*, *double*, *boolean*, *octet* e *char*. Literais estruturados têm uma estrutura de tupla, incluindo tipos como *date* e *time*. O

usuário pode definir literais estruturados como *needed* (usando o construtor *Struct*). Literais de coleção especificam uma coleção de objetos ou literais. Tipos de coleções são: *set*, *bag*, *list array* e *dictionary*. Cada coleção tem um grupo de operadores embutidos (Cattell et al., 2000; Badia, 2005).

Atributos complexos tornam-se atributos estruturados, e atributos multivalorados podem ser gerados com um construtor *Set*.

## 6.2. Gerar classes

Cada classe representada na UML é transposta para uma classe em ODL e cada associação identificada como um atributo de relacionamento com um inverso. Classes de associação podem ser tratadas como classes regulares, considerando que em ODL não se pode tratar diretamente relacionamentos com atributos. A herança é modelada diretamente tanto em UML quanto em ODL (Badia, 2005).

A ODL supõe que as classes são persistentes e a sintaxe que define essa persistência é específica de cada fabricante (Cattell et al., 2000).

Nessa etapa são geradas as classes, como exemplificado abaixo (figura 4a). A partir da especificação de uma superclasse, são geradas as classes especializadas, com as características herdadas da superclasse, e agregando suas próprias características. O padrão ODMG define a extensão de um tipo como sendo o conjunto de todas as instâncias do tipo em um determinado banco de dados. Com a ODL, basta especificar a palavra-chave de extensão e um nome, como o trecho abaixo (figura 4b):

Na ODL do modelo de objetos do ODMG 3.0, classes são declaradas através de uma interface (usando a palavra chave *interface*); cada classe é uma coleção de atributos, que são primitivos (de um tipo de dados básico) ou relacionamento (seu valor é um objeto ou conjunto de objetos de uma determinada classe).

Portanto, nessa etapa são gerados os modelos de *script*, segundo as referências da ODL, para o mapeamento do diagrama representado pela UML para BDOO. Cada classe representada na UML é transposta para uma classe em ODL e cada associação identificada como um atributo de relacionamento com um inverso. A herança é modelada diretamente em UML e em ODL. A ODL supõe que as classes são persistentes e a sintaxe que define essa persistência é específica de cada fabricante (Cattell et al., 2000).

Considerando a proposta do mapeamento do modelo de PosicaoConta, os primeiros elementos a serem mapeados são as classes fatos e dimensões. A superclasse Pessoa abrange características das classes Gerente e Cliente, com atributos gerais para as classes e específicos para cada uma delas separadamente, conforme o *script* em ODL da figura 4.

<pre>class Pessoa (extent Pessoas) {attribute string NomePessoa; attribute string ChavePessoa; ...;}</pre> <p style="text-align: right;">(a)</p>	<pre>class Cliente extends Pessoa {...} class Gerente extends Pessoa {...}</pre> <p style="text-align: right;">(b)</p>
--	--

Fig. 4. Superclasse Pessoa e classes Cliente e Gerente em ODL.

### 6.3. Gerar Identificadores e Descritores

As propriedades de identificação (OID) e descritor do objeto (D) são representadas através da utilização do diagrama de classes da UML, porém, ignorados no mapeamento, considerando que o OID é gerado automaticamente pelo BDOO e o descritor é uma propriedade lógica do modelo dimensional proposto. As regras de derivação de atributos e aditividade não são especificadas diretamente pela ODL, portanto, devem ser consideradas a partir dos conceitos definidos e implementados através das regras da OO.

### 6.4. Gerar Relacionamentos

Em ODL, um relacionamento é definido explicitamente pela declaração de caminhos transversos que permitem aplicações que usam conexões lógicas entre os objetos participantes do relacionamento. Através da declaração Set pode-se indicar uma coleção não-ordenada de elementos sem duplicatas, usado para definir relacionamentos entre classes. O relacionamento de agregação não é representado diretamente em ODL, portanto, deve ser transformado em um relacionamento de associação.

A vantagem em uma associação simples em UML advém do encapsulamento e da conseqüente limitação de acoplamento das classes relacionadas. Se a multiplicidade for 0..1 ou 1..1 a ODL provê a declaração de um atributo de um tipo de classe, ocultando a implementação da estrutura de dados. A literal estruturada é um tipo de literal suportada pelo modelo de objetos. Além dos tipos de estrutura suportados pela ODMG, que são date, interval, time e timestamp, pode-se definir outras estruturas como extensão dos modelos de objeto (Cattell et al., 2000).

No modelo proposto, observa-se que cada cliente pode ter várias contas, bem como cada conta pode conter vários produtos, gerando uma tabela bridge entre as tabelas Conta-Cliente e entre Conta-Produto. Portanto, a figura 5a mostra em ODL a geração da classe ContaCliente. Nas classes Conta e Cliente são gerados os relacionamentos correspondentes, em ODL (figura 5b e figura 5c).

<pre>Class ContaCliente (extent CtaCli) { relationship set&lt;Conta&gt; ItemCta inverse Conta::contas; relationship set&lt;Cliente&gt; ItemCli inverse Cliente::clientes;}</pre>	(a)
<pre>class Conta (extent Contas) {...relationship set&lt;ContaCliente&gt; contas inverse ContaCliente:: ItemCta; attribute date DataAbertura; ...}</pre>	(b)
<pre>class Cliente (extends Pessoa) {... relationship set&lt;ContaCliente&gt; clientes inverse ContaCliente::ItemCli; attribute long CPFCliente; ...}</pre>	(c)

Fig. 5. Definição de relacionamento n:n em ODL.

## 7. Etapa 5: Implementar o Modelo

Na modelagem dimensional, os projetos lógicos e físicos são muito semelhantes e as diferenças apresentam-se em termos de detalhes especificados para o banco de dados

físico, incluindo nomes de coluna, tipos de dados, declarações de chave e a permissibilidade de nulos (Kimball, 2002). No projeto físico observam-se ainda elementos básicos como ajuste de desempenho, particionamento e layout do arquivo, porém, que não fazem parte do escopo desse artigo.

Para a validação da proposta será utilizado o banco de dados pós-relacional Caché – Intersystems. O modelo de objetos do Caché é baseado no padrão ODMG. Além disso, suporta um conjunto de conceitos de programação por objeto que inclui encapsulamento, objetos embutidos, herança múltipla, polimorfismo e coleções. O Caché gerencia o OID através de mecanismos próprios, atribuindo automaticamente para cada nova instância de objeto criado um OID, na forma de um número inteiro sequencial.

Para o banco de dados Caché um relacionamento é uma associação entre dois objetos, de um tipo específico. Para criar um relacionamento entre dois objetos, cada um deve ter uma propriedade Relationship, que define sua parte do relacionamento. Observando que as classes são definidas como persistentes, para armazenamento definitivo dos dados, utilizam-se conceitos de herança, herdando os métodos da classe Persistent. Considerando a superclasse Pessoa uma generalização das classes Gerente e Cliente, com atributos gerais para a superclasse e específicos para as classes especializadas, o *script* da figura 6 define tais características.

<pre>Class modelo.Pessoa Extends (%Persistent) [ ClassType = persistent, ProcedureBlock ] {Property...}</pre> <p style="text-align: right;">(a)</p>	<pre>Class modelo.Cliente Extends modelo.Pessoa [ ClassType = persistent, ProcedureBlock ] {Relationship PosicaoCliente As modelo.Posicao [ Cardinality = many, Inverse = Cliente ]; ...}</pre> <p style="text-align: right;">(b)</p>
---	---

Fig. 6. Classe Cliente herda características da classe Pessoa.

Os Fatos também são mapeados para classes. As classes dimensão Conta e Cliente caracterizam um relacionamento n:n, gerando a classe bridge de associação ContaCliente, conforme o *script* abaixo gerado para as classes Conta (na figura 7a), Cliente (figura 7b) e ContaCliente (na figura 7c), respectivamente:

<pre>Class modelo.Conta Extends Extends %Persistent [ ClassType = persistent, ProcedureBlock ] {Relationship PosicaoConta As modelo.Posicao [ Cardinality = many, Inverse = Conta ]; ...}</pre> <p style="text-align: right;">(a)</p>
<pre>Class modelo.Cliente Extends modelo.Pessoa [ ClassType = persistent, ProcedureBlock ] {Relationship PosicaoCliente As modelo.Posicao [ Cardinality = many, Inverse = Cliente ]; ...}</pre> <p style="text-align: right;">(b)</p>
<pre>Class modelo.ContaCliente Extends %Persistent [ ClassType = persistent, ProcedureBlock ] {Relationship Conta As modelo.Conta [ Cardinality = parent, Inverse = ItemCta ]; Relationship Cliente As modelo.Cliente [ Cardinality = one, Inverse = ItemCli ]; ...}</pre> <p style="text-align: right;">(c)</p>

Fig. 7. Classes Conta, Cliente e ContaCliente.

## **8. Considerações Finais**

A proposta metodológica aqui apresentada centra-se na aplicação do modelo dimensional modelado através da UML e implementado em um banco de dados orientado a objetos, utilizando a persistência de dados.

Muitos produtos focam seus esforços na tecnologia objeto-relacional, que não implementa totalmente os conceitos da OO, como a herança múltipla, por exemplo. Esses conceitos são implementados segundo a abordagem relacional, gerenciados na interface com o usuário, exigindo um mapeamento da definição orientada a objetos para o modelo relacional, implicando em se refazer um trabalho já modelado.

Com a utilização de um modelo de BDOO, as características e conceitos da orientação serão preservados e aplicados segundo sua definição e modelagem, não exigindo que o trabalho de modelagem seja refeito e seu conseqüente mapeamento.

Portanto, o trabalho apresentou uma metodologia para a implantação do ambiente de DW, caracterizado pelo modelo multidimensional segundo o paradigma da OO. A representação do modelo através do diagrama de classes, a partir do modelo ER, fornece um ambiente natural do modelo OO, para posterior implementação. O mapeamento é feito naturalmente, seguindo-se as definições do padrão da OO – ODMG.

## **Referências Bibliográficas**

- Abelló, A.; Samos, J.; Saltor, F. (2005). "YAM<sup>2</sup>: a multidimensional conceptual model extending UML". Information Systems, In Press, Corrected Proof, Available online.
- Badia, A. (2005). "From Conceptual Models to Data Models". In P. Van Bommel, Transformation of Knowledge, Information and Data: Theory and Applications (pp. 148-170), Hershey, PA: Information Science Publishing.
- Cattell, R. G. G.; Barry, D. K.; Berler, M. (2000). "The object data standard: ODMG 3.0". San Francisco: Morgan Kaufmann Publishers.
- Kimball, R. (1997). "A dimensional modeling manifesto". <<http://www.rkimball.com/html/articlesArchitecture.html>>.
- \_\_\_\_\_. (2002). "Data Warehouse toolkit: o guia completo para modelagem multidimensional". Rio de Janeiro: Campus.
- Lee, H. J.; Lee, S. W.; Kim, H. J. (2005). "Design and implementation of an extended relationship semantics in an ODMG-compliant OODBMS". The Journal of Systems and Software, 76.
- Muller, R. J. (2002). "Projeto de Banco de dados: usando UML para modelagem de dados". São Paulo: Berkeley Brasil.
- Nassu, E. A.; Setzer, V. W. (1999). "Bancos de dados orientados a objetos". São Paulo: Edgard Blücher Ltda.
- ODMG. (2005). "Object Data Management Group". <<http://www.odmg.org>>.
- Shin, S. K.; Sanders, G. L. (2005). "Denormalization strategies for data retrieval from data warehouses". Decision Support Systems, In Press, Corrected Proof.

- Trujillo, J., Lujan-Mora, S.; Song, I.-Y. (2003) "Applying UML for Designing Multidimensional Databases and OLAP Applications". In K. Siau (Ed.), *Advanced Topics in Database Research, Volume 2* (pp. 13-36), Hershey, PA: Idea Group Publishing.
- Turowski, K. (2000) "Establishing Standards for Business Components". In K. Jakobs (Ed.) *Information Technology Standards and Standardization: A Global Perspective* (pp. 131-151), Hershey, PA: Idea Group Publishing.